REMARKS

This application has been carefully considered in connection with the Examiner's Final Office Action dated January 25, 2008. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 1 – 32 and 47-53 and 55-69 were pending at the time of the Final Office Action.

Claims 1–16, 19-32, 47-53 and 55-60 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller et al., U.S. Patent No. 5,754,855 (hereinafter "Miller") in view of E. Reed Doke; Bill C. Hardgrave, (1998 John Wiley & Sons, Inc., "An Introduction to Object Cobol" (hereinafter "Doke").

Claims 17 and 18 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke and further in view of Keith Haviland, Dina Gray and Ben Salama, Pearson education Limited, 1998 "Unix system programming, second edition" (hereinafter "Haviland").

Summary of Response

Claims 1, 5-10, 20, and 55-69 were previously presented.

Claims 2-4, 11-19, 21-32, and 47-53 remain as originally filed.

Claims 33-46 and 54 were previously canceled.

Summary of Claims Pending

Claims 1-32, 47-53, and 55-69 are currently pending following this response.

Improper Final Rejection

Applicant respectfully submits that the finality of the Final Office Action dated January 25, 2008 is improper. MPEP 706.07(a) states, "second or any subsequent actions on the merits shall be final, **except where the examiner introduces a new ground of rejection that is neither necessitated by applicant's amendment of the claims, nor based on information submitted in an information disclosure statement**" (emphasis added). Applicant respectfully submits that the new ground of rejection introduced by the Examiner against at least independent Claims 20 and 47 was neither necessitated by Applicant's amendment of the claims, nor based on information submitted in an information disclosure statement. In the Office Action dated July 26, 2007, Claims 20 and 47 were rejected under 35 U.S.C. 103(a) by Miller in view of Obin. In the Final Office Action dated January 25, 2008, Claims 20 and 47 were rejected under 35 U.S.C. 103(a) by Miller in view of Doke. Applicant notes that Claims 20 and 47 were not amended in the response filed on October 26, 2007. Applicant further notes that claim 47 remains as originally presented. Also, no information disclosure statement has been filed citing the Doke reference prior to the Final Office Action. Accordingly, Applicant respectfully submits that the finality of the Final Office Action dated January 25, 2008 is premature and respectfully requests the pending application be withdrawn from final rejection.

Response to Rejections

The disclosure is related to implementing distributed and asynchronous processing in COBOL. As disclosed in paragraphs 006-008 of the disclosure, programming languages such as C and JAVA have functionality for performing distributed and asynchronous processing, such as

shared memory and message queues, threads, semaphores and mutexes, events, signal handlers, and sockets. The distributed and asynchronous processing functionality available in C and JAVA was unavailable in COBOL. Because many businesses have legacy applications that have been developed in COBOL that are well suited for performing their intended tasks, it is difficult for businesses to abandon those applications. Some solutions to abandoning the legacy COBOL application is to redeveloping the COBOL applications in C or JAVA or the legacy COBOL applications may be provided with an interface to cooperate with C or JAVA programs. When the legacy COBOL applications are provided with the interface, the C or JAVA programs may then perform the distributed and asynchronous processing tasks that may be necessary in modern business environments.

Rather than redeveloping the COBOL applications or utilizing an interface with another programming language, such as C or JAVA, the disclosure enables COBOL applications to perform distributed and asynchronous processing tasks through a technical layer using functionality native to COBOL. Paragraphs 027-039 of the disclosure provide a detailed description of the technical layer. For example, the technical layer may be COBOL modules, routines, or paragraphs that may be defined within a COBOL library. The COBOL library may then be linked into a COBOL program, such that the COBOL program may utilize the functionality of the technical layer through calls to the COBOL modules, routines, or paragraphs. The claims are directed to the distributed and asynchronous functionality enabled in a COBOL program through the disclosed technical layer.

Miller is directed to generating a program with routines written using code segments that are computer language diverse and/or programming paradigm diverse. Miller discloses in column 7, lines 11-55 to enable communication between the diverse routines event tokens may be used. Miller discloses in column 7, lines 42-45, "All application programs 126 and the functions which make up the application programs 126 are required to conform to the syntax associated with event tokens." Therefore, Miller enables communication between diverse programming routines by requiring the programming routines to conform to a particular syntax when communicating. Therefore, the disclosure of Miller may use a COBOL routine for performing COBOL functions and use a C routine for performing POSIX functions or other non-native COBOL functions similar to other prior art approaches described above.

Doke is directed to Object COBOL which uses object oriented programming techniques in the procedurally oriented COBOL programming language. Object COBOL enables writing COBOL code to define classes that describe attributes and define methods for a class. Instances of the class may be invoked in a driver program written in COBOL. Accordingly, Doke discloses utilizing COBOL code and object orient programming techniques to write object oriented programs. Doke does not provide any teaching or suggestion of using Object COBOL to perform POSIX functions.

In light of the differences between the goals of the disclosure and the cited art, the differences between the claim limitations and the cited portions of the prior art are discussed in detail below.

Response to Rejections under Section 103

Claims 1–16, 19, 21-32, 48-53 and 56-60 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Claim 1:

Claim 1 was rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

I. Miller does not teach or suggest an index including a process identified associated with a child process.

Claim 1 recites, "maintaining, in a COBOL program, an index including a process identifier and an event associated with a child process."

The Final Office Action relied on disclosure in Miller of a stack frame of a COBOL routine to read on this limitation. Miller discloses in column 5, lines 40-46, "An invocation stack is a last-in-first-out (LIFO) stack (which may be implemented as a linked list). The elements of an invocation stack are called stack frames[.] Each thread of an application program is associated with an invocation stack. Each routine of a thread is associated with a stack frame in the invocation stack." Miller also discloses in column 5, line 57, "Stack frame 306 is associated with the COBOL routine 128." Miller further discloses in column 6, lines 5-9, "A routine may among other things use its stack frame to store dynamic variables. Also, when one routine ... calls another routine ..., the state of the calling routine is stored in the calling routine's stack frame." From the cited disclosure of Miller, it is clear that a stack frame is a temporary storage location that stores dynamic variables of a routine while another routine operates. Accordingly, a stack frame is not a child process as implied by the Final Office Action. Also, a stack frame is not a process

identifier and does not include a process identifier. Further, a stack frame is not an index maintained by a COBOL program that includes a process identifier associated with a child process, as claimed.

MPEP 2173.05(a)(III) states:

> "In applying the prior art, the claims should be construed to encompass all definitions that are consistent with applicant's use of the term. See *Tex. Digital Sys., Inc. v. Telegenix, Inc.*, 308 F.3d 1193, 1202, 64 USPQ2d 1812, 1818 (Fed. Cir. 2002). It is appropriate to compare the meaning of terms given in technical dictionaries in order to ascertain the accepted meaning of a term in the art. In re Barr, 444 F.2d 588, 170 USPQ 330 (CCPA 1971). >See also MPEP § 2111.01.<"

Applicant respectfully submits that the term "child process" is a term of art. The online wiki encyclopedia Wikipedia found at the URL, http://en.wikipedia.org/wiki/Child_process, defines the term "child process" as "a computer process created by another process (the parent process)." Further, paragraph 090 of the specification discloses, "The child processes are subtasks, subprograms, or subroutines of the parent program." Accordingly, based on the understanding of one skilled in the art at the time of the invention, it is clear that a stack frame is not a child process.

Also, as noted in the response filed on October 26, 2007, Applicant respectfully submits that the term "process identifier" is a term of art. The online technical dictionary TechWeb found at the URL http://www.techweb.com/encyclopedia/ defines the term process identifier (PID) as, "A temporary number assigned by the operating system to a process or service." Accordingly, based on the understanding of one skilled in the art at the time of the invention, it is clear that a stack frame is not a process identifier and does not include a process identifier.

Therefore, based on the understanding of the terms "child process" and "process identifier" by one skilled in the art at the time of the invention, it is clear that a stack frame is not an index

maintained by a COBOL program that includes a process identifier associated with a child process, as claimed.

II. Miller does not teach or suggest placing the child process in a wait state when the child process is initialized.

Claim 1 recites, "placing the child process in a wait state when the child process is initialized."

The Final Office Action relied on disclosure of synchronous signaling in column 6, lines 58-63 to read on this limitation. Miller discloses in column 6, lines 54-63, "A synchronous signal is an indication of the occurrence of an event whose origin can be attributed to a specific thread ..., wherein the originating thread and the receiving thread ... execute synchronously with respect to processing the signal. In other words, with synchronous signals, the execution of the originating thread is suspended while the receiving thread is processing the signal." Also, as noted above, Miller discloses in column 6, lines 5-9, "[W]hen one routine ... calls another routine ..., the state of the calling routine is stored in the calling routine's stack frame." Accordingly, Miller discloses when performing synchronous processing between threads or routines, the thread or routine that performs the calling is placed in a wait state and the thread or routine that is being called actively processes the call. Quite the opposite, the claims require that when the child process is initialized by the COBOL program, the child process is placed in a wait state, not the COBOL program.

III. Miller does not teach or suggest signaling, by the COBOL program, the child process to run using the process identifier and the event associated with the child process.

Claim 1 recites, "signaling, by the COBOL program, the child process to run using the process identifier and the event associated with the child process."

The Final Office Action again relied on disclosure of synchronous signals in Miller to read on this limitation. Applicant respectfully submits that the use of synchronous signals does not provide any teaching or suggestion of signaling a child process to run using a process identifier and an event associated with the child process, as claimed. Applicant notes that Miller discloses in Fig. 2 and column 5, lines 49-51, "As shown in this routine invocation sequence 202, the COBOL routine 128 has invoked the PL/I routine 130, and the PL/I routine 130 has invoked the C routine 132." Accordingly, while Miller discloses that the COBOL routine may invoke the PL/I routine, Miller does not provide any teaching or suggestion that the COBOL routine uses a processes identifier and an event.

IV. Doke does not cure the deficiencies of Miller.

The Final Office Action concedes, "Miller does not explicitly teach an index including a process identifier; and initializing, by the COBOL program, the child process." The Final Office Action relied on disclosure of factory methods and instance methods of Object COBOL to read on these limitations. Applicant respectfully submits that a search of Doke for the term "process identifier" did not produce any results. Accordingly, neither Miller nor Doke teach or suggest a process identifier. Further, Applicant notes that the factory methods and instance methods of Doke are disclosed to be two types of methods that may be defined in a class program (see page 39 of Doke). Doke discloses on page 45, "An instance method is used to do processing for individual instances (individual accounts in our example). A factory method handles the processing for a group of objects (all accounts in our example)." As disclosed on page 42 and described in the examples of Figs. 5.3, 5.9, and 5.11, driver programs may be used to invoke methods of a class program using the INVOKE verb. Accordingly, child processes are not initialized by a COBOL

program as claimed. Rather, class programs are invoked by driver programs. As described in paragraph 092 of the pending disclosure, "initializing or on starting the child process."

Assuming arguendo, even if the disclosure in Doke of invoking a class program is disclosure of initializing a child process, Doke does not teach or suggest that the child process is placed in a wait state when it is initialized, as claimed. Rather, Doke discloses in at least pages 42 and 43 that when a method of a class program is invoked, it is executed (i.e., the invoked method does not wait to run).

For at least the reasons established above in sections I-IV, Applicant respectfully submits that independent Claim 1 is not taught or suggested by Miller in view of Doke and respectfully requests allowance of this claim.

Claims Depending From Claim 1:

Claims 2–16 and 19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Claims 17 and 18 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke and further in view of Haviland.

Dependent Claims 2-19 depend directly or indirectly from independent Claim 1 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-IV above, Applicant respectfully submits that Claims 2-19 are not taught or suggested by Miller in view of Doke and respectfully requests allowance of these claims. Applicant respectfully submits that Haviland does not cure the deficiencies of Miller and Doke noted above.

Claim 20:

Claim 20 was rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Claim 20 was rejected with reference to the limitations of Claim 1. Applicant respectfully submits that the limitations of Claim 20 and independent Claim 1 are different. Accordingly, Applicant respectfully submits that the Final Office Action has failed to establish a *prima facie* case of obviousness against Claim 20 by not providing a finding that the prior art as a whole in one form or another contains all of the elements of Claim 20. In particular the Final Office Action has not provided a finding for the elements of Claim 20 that are different than the elements of Claim 1.

V. Miller in view of Doke doe not teach or suggest a module callable by the first and second COBOL programs, the module maintaining a state sharable between the COBOL programs to coordinate their processing.

Claim 20 recites, "a module recorded on a computer-readable medium callable by the first and second COBOL programs, the module maintaining a state shareable between the first and second COBOL programs to coordinate the processing of the first and second routines."

The Final Office Action did not address this limitation. Applicant notes that Miller only discloses a single COBOL routine. Accordingly, Miller does not provide any teaching or suggestion of a module that maintains a state sharable between the first and second COBOL programs, as claimed. Doke does not cure the deficiencies of Miller. While Doke may disclose two COBOL programs, Doke does not provide any teaching or suggestion of a module callable by

the first and second COBOL programs, where the module maintains a state sharable between the first and second COBOL programs, as claimed.

As disclosed in Fig. 5 and paragraphs 059-067 of the specification, the disclosed technical layer may provide support for semaphores and mutexes with COBOL programs. As noted above, COBOL does not provide native support for POSIX functionality such as semaphores and mutexes. Accordingly, COBOL programs have not previously been able to perform coordinated processing on shared resources between two or more COBOL programs. As disclosed by Applicant's specification, the semaphore routine 20i or the mutex routine 20j may be used to support semaphores and mutexes for maintaining a state sharable between two COBOL programs to enable coordinated processing of a shared resource. As described above, none of the applied art teaches or suggests such features.

For at least the reasons established above in section V, Applicant respectfully submits that independent Claim 20 is not taught or suggested by Miller in view of Doke and respectfully requests allowance of this claim.

Claims Depending From Claim 20:

Claims 21-32 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Dependent Claims 21-32 depend directly or indirectly from independent Claim 20 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in

section V above, Applicant respectfully submits that Claims 21-32 are not taught or suggested by Miller in view of Doke and respectfully requests allowance of these claims.

Claim 47:

Claim 47 was rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Claim 47 was rejected with reference to the limitations of Claim 1. Applicant respectfully submits that the limitations of Claim 47 and independent Claim 1 are different. Accordingly, Applicant respectfully submits that the Final Office Action has failed to establish a *prima facie* case of obviousness against Claim 47 by not providing a finding that the prior art as a whole in one form or another contains all of the elements of Claim 47. In particular the Final Office Action has not provided a finding for the elements of Claim 47 that are different than the elements of Claim 1.

VI. Miller in view of Doke does not teach or suggest registering, by a COBOL language program, a signal handler with an operating system.

Claim 47 recites, "registering, by a COBOL language program, a signal handler with an operating system, the signal handler associated with an event."

While Miller discloses in column 11, lines 53-59 that event handlers for stack frames may be registered with an event manager 124 through a registration event manager 604, Miller does not disclose that the event handlers are registered by a COBOL program or that the event handlers are registered with an operating system as claimed. Applicant respectfully submits that Doke does not cure this deficiency of Miller.

For at least the reasons established above in section VI, Applicant respectfully submits that independent Claim 47 is not taught or suggested by Miller in view of Doke and respectfully requests allowance of this claim.

Claims Depending From Claim 47:

Claims 48-53 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Dependent Claims 48-53 depend directly or indirectly from independent Claim 47 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in section VI above, Applicant respectfully submits that Claims 48-53 are not taught or suggested by Miller in view of Doke and respectfully requests allowance of these claims.

Claim 55:

Claim 55 was rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Claim 55 includes limitations substantially similar to the limitations discussed in sections I-IV above. For at least the reasons established above in sections I-IV, Applicants respectfully submit that independent Claim 55 is not taught or suggested by Miller in view of Doke and respectfully request allowance of this claim.

Claims Depending From Claim 55:

Claims 56-60 were rejected under 35 U.S.C. 103(a) as being unpatentable over Miller in view of Doke.

Dependent Claims 56-69 depend directly or indirectly from independent Claim 55 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I-IV above, Applicant respectfully submits that Claims 56-69 are not taught or suggested by Miller in view of Doke and respectfully requests allowance of these claims.

Conclusion

Applicant respectfully submits that the present application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, he is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: March 24, 2008

/Michael W. Piper/
Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANT

CONLEY ROSE, P.C.
5601 Granite Parkway, Suite 750
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)